HW4

Github link: https://github.com/sabbellasri/Deep-learning/tree/main/hw4

Introduction

In this project, we're exploring Generative Adversarial Networks (GANs), a powerful class of unsupervised learning models. GANs operate on the principle of competition between two neural networks: the generator and the discriminator. The generator's role is to produce synthetic data samples, while the discriminator's task is to differentiate between real and fake data.

Unlike traditional programming approaches where outputs are explicitly defined, GANs learn directly from the data they're exposed to, allowing them to generate new data that closely resembles the training data. This adaptability makes GANs invaluable for tasks such as image, video, and voice generation.

Here's the gist of how GANs function: the generator takes random noise as input and generates data samples, aiming to mimic the distribution of the training data. The discriminator then evaluates these samples, attempting to distinguish between real and synthetic data. Through an adversarial training process, both networks continually improve: the generator becomes more proficient at generating realistic data, while the discriminator enhances its ability to discern real from fake.

Problem Statement

Train a discriminator/generator pair on the CIFAR10 dataset utilizing techniques from DCGAN, Wasserstein GANs, and ACGAN.

Deep Convolutional Generative Adversarial Networks (DCGANs) are a specialized form of convolutional neural networks (CNNs) designed specifically for unsupervised learning tasks, with a particular emphasis on image generation. DCGANs stand out for their unique architectural constraints, which prioritize convolutional layers over fully connected layers.

This architectural choice, along with techniques such as strided convolutions for downsampling and transposed convolutions for upsampling, enhances the model's adaptability and effectiveness in generating images. DCGANs have revolutionized the field by demonstrating that increasing the complexity of the generator doesn't automatically lead to better image quality. Instead, meticulous attention to architectural design principles is essential for successful training.

Key guidelines for DCGAN architecture include replacing max-pooling layers with convolutional strides, utilizing transposed convolutions for upsampling, and eliminating fully connected layers. These principles optimize the network's ability to learn meaningful representations from the data and generate high-fidelity images.

To address training challenges and prevent issues like model collapse, batch normalization is often integrated into the network architecture. Additionally, activation functions such as LeakyReLU are commonly employed in the generator, while the discriminator typically utilizes the hyperbolic tangent (tanh) function for its output.

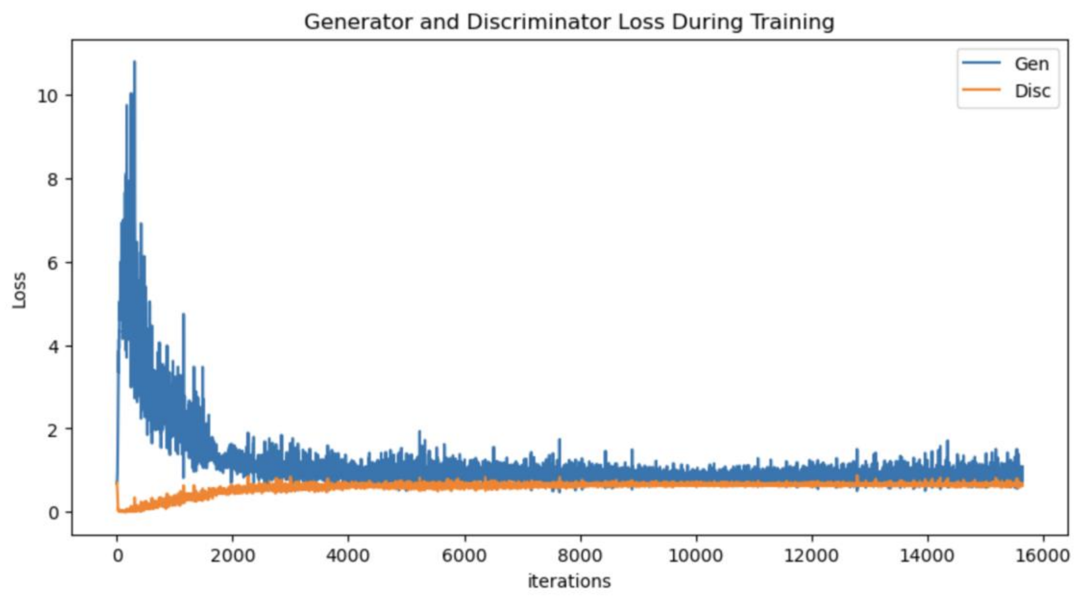
The network was trained for 40 epochs with the following, hyperparameter configuration: o RATE= RATE OF LEARNING 5e-5

BT_SIZE=128
IMG_SIZE = 64, to resize the image.
CHANNELS_IMG=3
NOISE_DIM=100
NUMBER OF EPOCHS=40
FEATURES_DISC=64

FEATURES_GEN=64
Adam Optimiser

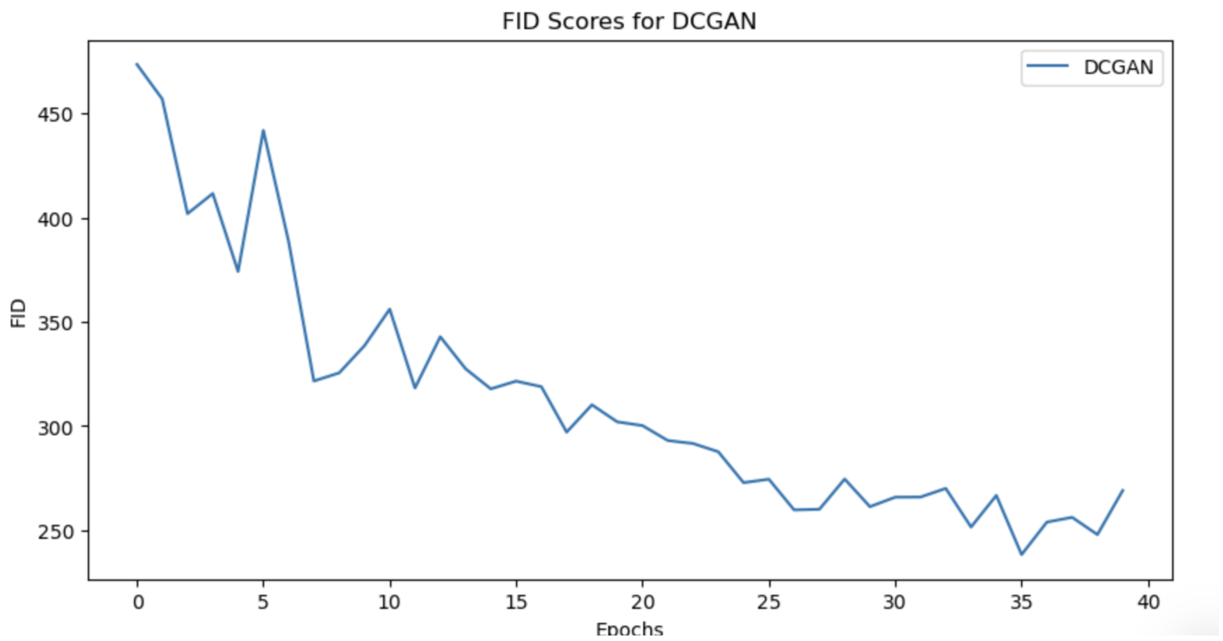
Momentum=0.9,
beta=0.5

After 40 epochs the loss graph is

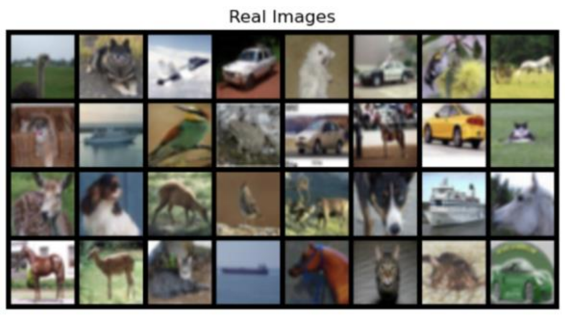


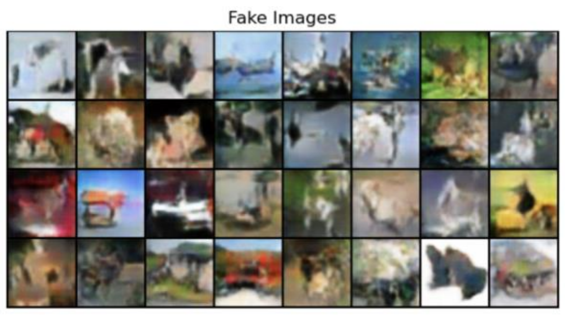
In approximately 2500 iterations, both the generator and discriminator loss stabilize, maintaining a consistent ratio, which is indicative of an ideal condition.

FID Scores for DCGAN Epoch wise



Experimenting with different epochs and learning rates revealed that beyond the 55th epoch, the generator loss tends to increase, resulting in a decline in the quality of the generated images.





Here are the images generated at epoch number 32, which achieved the lowest FID score.

ACGAN

The Auxiliary Classifier GAN, or AC-GAN, represents an advancement of the conditional GAN (CGAN) architecture. Introduced in the 2016 paper "Conditional Image Synthesis using Auxiliary Classifier GANs" by Augustus Odena et al. from Google Brain, AC-GAN builds upon the foundation laid by CGAN.

Similar to CGAN, the AC-GAN's generator model is conditioned on both a point in the latent space and a class label, enabling conditional image generation. However, the discriminator model in AC-GAN differs significantly from its CGAN counterpart. While CGAN's discriminator takes both the image and the class label as input, AC-GAN's discriminator is only fed the image.

In AC-GAN, the discriminator's task is twofold: it must discern whether the input image is real or fake, as well as predict the class label associated with the image. By incorporating this auxiliary classifier into the discriminator, AC-GAN enhances the discriminative power of the model while simultaneously enabling it to perform class prediction tasks.

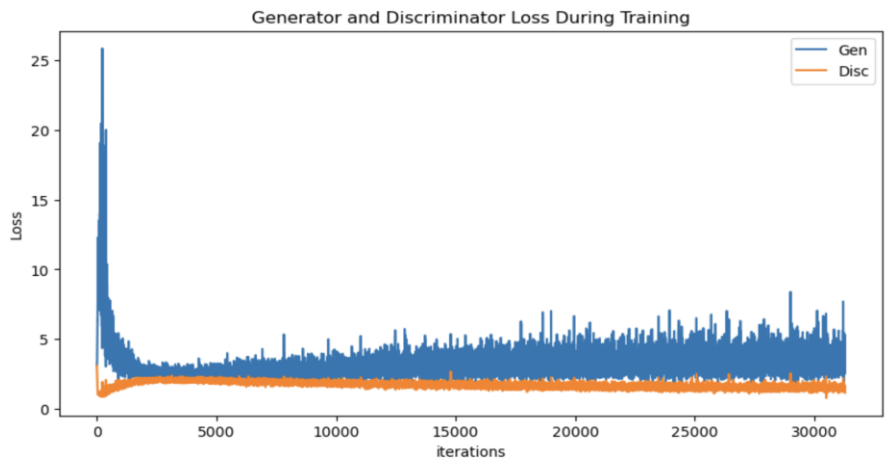
The network was trained for 40 epochs with the following, hyperparameter configuration: o RATE= RATE OF LEARNING 5e-5

BT_SIZE=128
IMG_SIZE = 64, to resize the image.
CHANNELS_IMG=3
NOISE_DIM=100
NUMBER OF EPOCHS=40
FEATURES_DISC=64

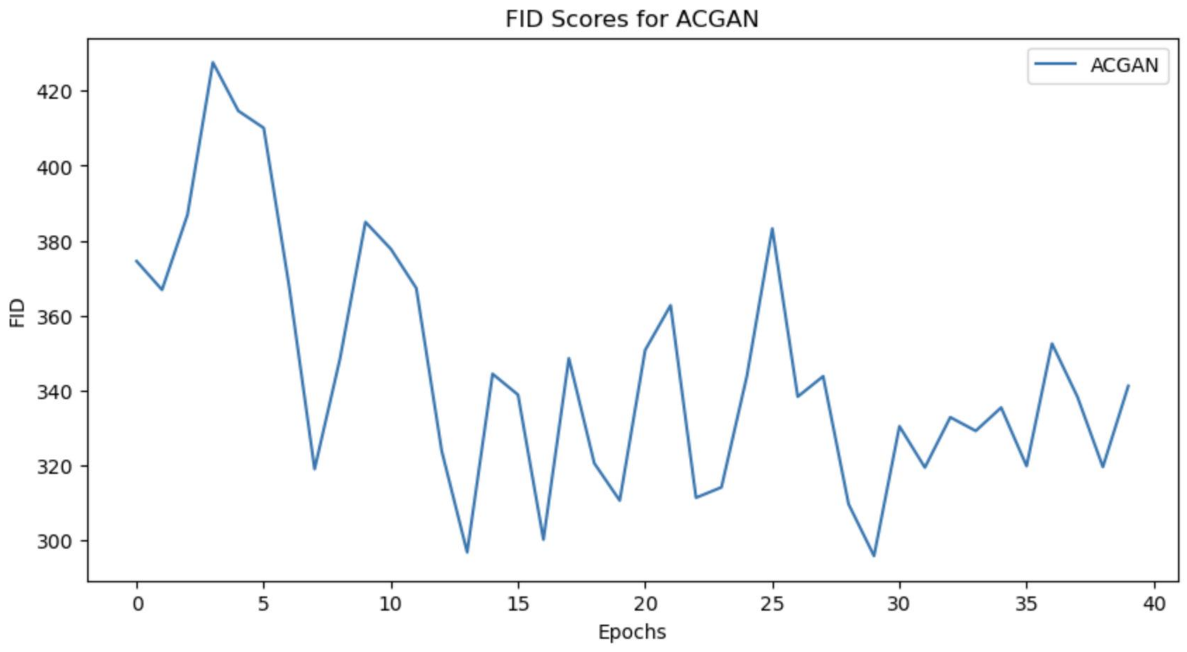
FEATURES_GEN=64
 Adam Optimiser

Momentum=0.9,
beta=0.5

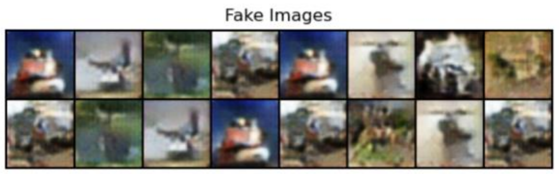
After 40 epochs the loss graph is



FID Scores for DCGAN Epoch wise



Here are the images generated at epoch number 32, which achieved the lowest FID score.



WCGAN

The Wasserstein GAN + Gradient Penalty, known as WGAN-GP, represents a refinement of the standard Wasserstein GAN (WGAN) architecture, aimed at addressing issues associated with weight clipping while ensuring Lipschitz continuity.

In the original WGAN, weight clipping is employed to enforce 1-Lipschitz continuity, but this approach can lead to undesirable effects such as pathological value surfaces, underutilization of network capacity, and problems with gradient explosion or vanishing.

WGAN-GP introduces a Gradient Penalty as an alternative method for enforcing the Lipschitz constraint. This penalty encourages the gradients of the discriminator network to have a consistent norm, thereby promoting smoother learning dynamics and mitigating the need for aggressive weight clipping.

Specifically, the squared difference from a target norm, typically set to 1, is used as the gradient penalty. This softer form of the Lipschitz requirement offers a more stable training process and avoids the pitfalls associated with weight clipping.

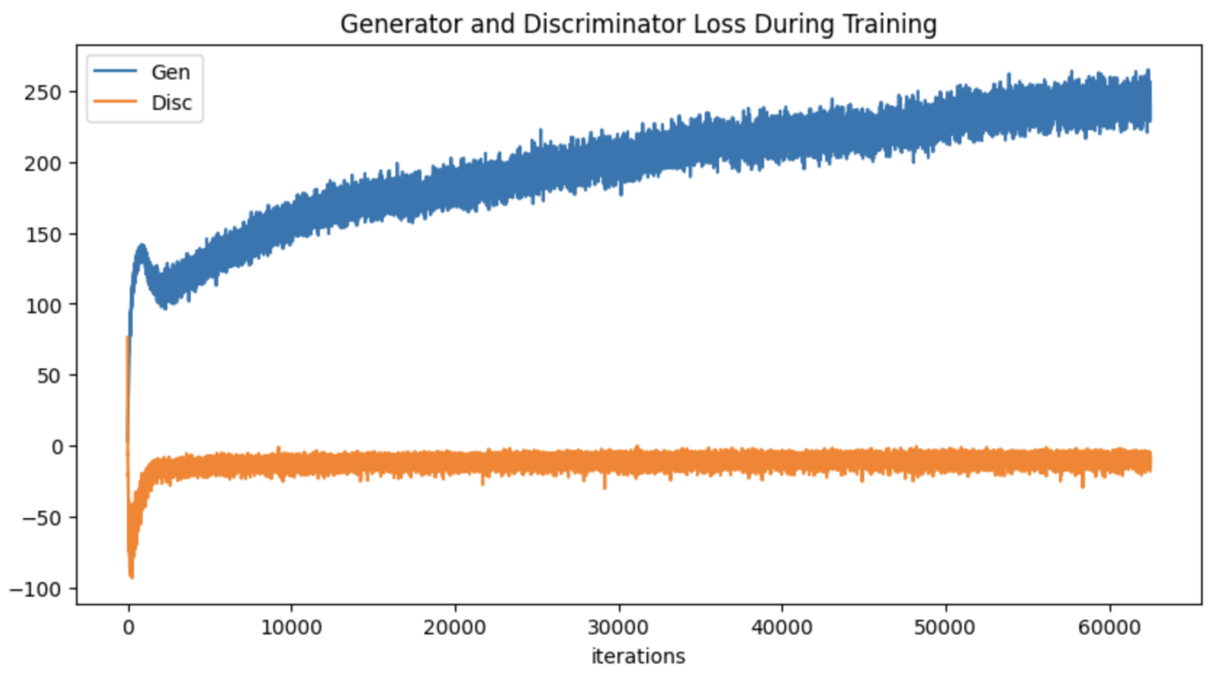
The network was trained for 40 epochs with the following, hyperparameter configuration

BT_SIZE=128
IMG_SIZE = 64, to resize the image.
CHANNELS_IMG=3
NOISE_DIM=100
NUMBER OF EPOCHS=40
FEATURES_DISC=64

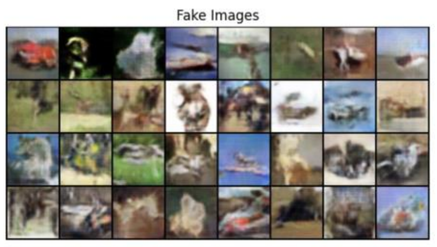
FEATURES_GEN=64
Adam Optimiser

Momentum=0.9,
beta=0.5

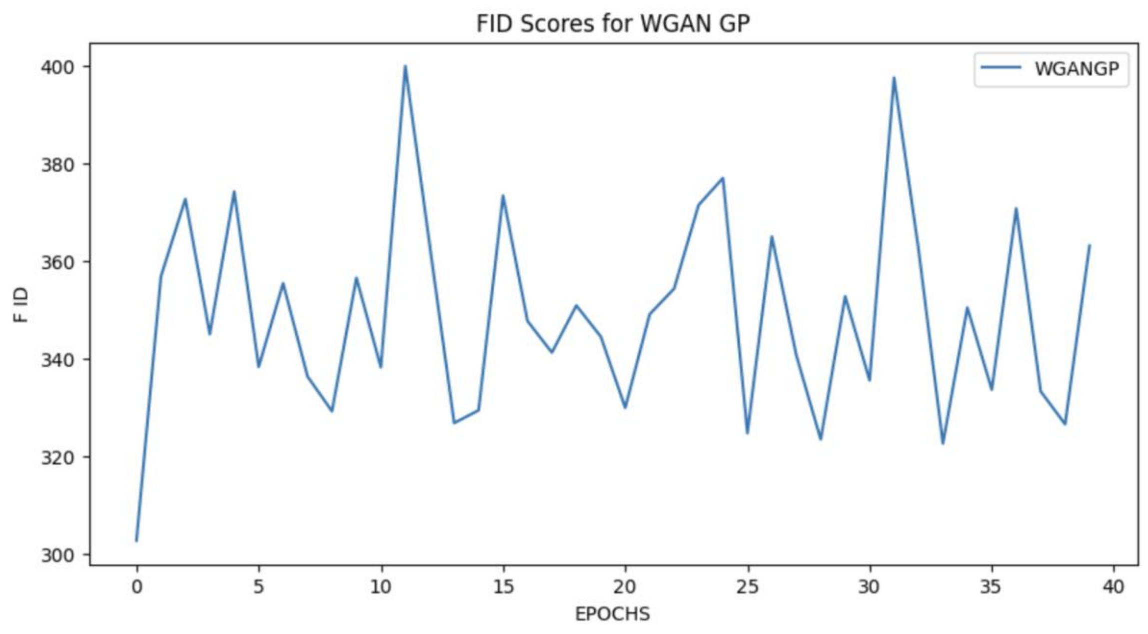
After 40 epochs the loss graph is



Here are the images generated at epoch number 32, which achieved the lowest FID score.



FID Score EPOCH Wise



Evaluation

The evaluation of the implemented GANs and their generated images relies on the Frechet Inception Distance (FID) score. This metric quantifies the dissimilarity between feature representations extracted from real images and those generated by the GAN.

By leveraging the Inception v3 image classification model, the FID score captures the statistical differences between the feature vectors of real and synthetic images, specifically focusing on aspects relevant to computer vision. A lower FID score indicates that the statistical properties of the two sets of images are more similar, suggesting higher quality and realism in the generated images. Conversely, a perfect FID score of 0.0 would imply complete identity between the distributions of real and generated images.

Model, FID Score (Min), FID Score (Mean), FID Score (Max)

DCGAN 238.36, 312.75, 473.5
ACGAN 295.8, 345.084, 427.54

As observed in the above statistics, DCGAN has the best performance of all the models.